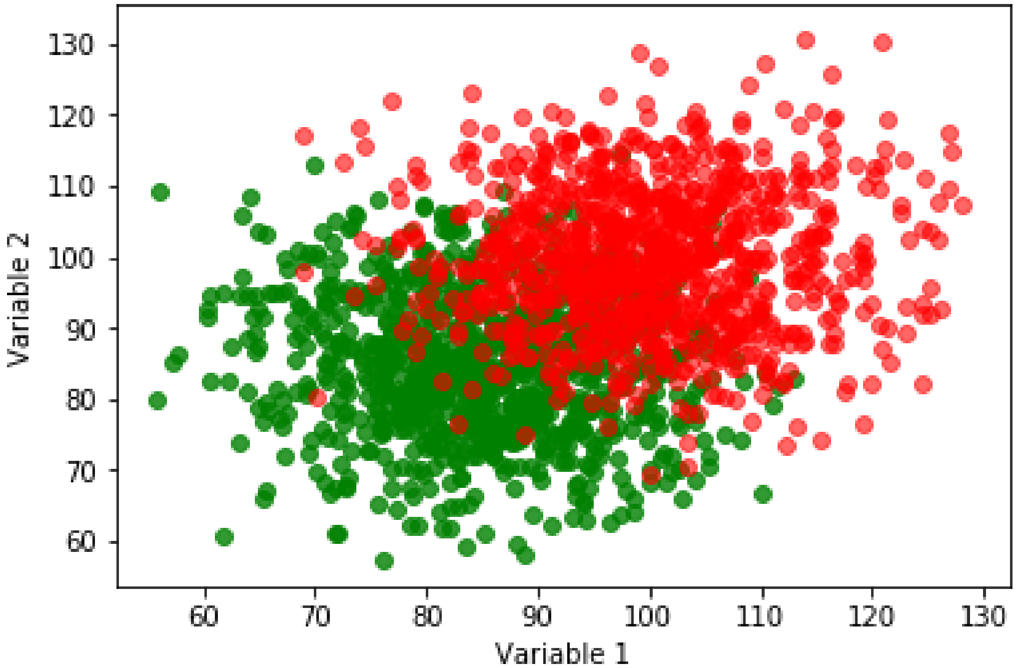
Normal_random_variates_XYScatterPlt (2)

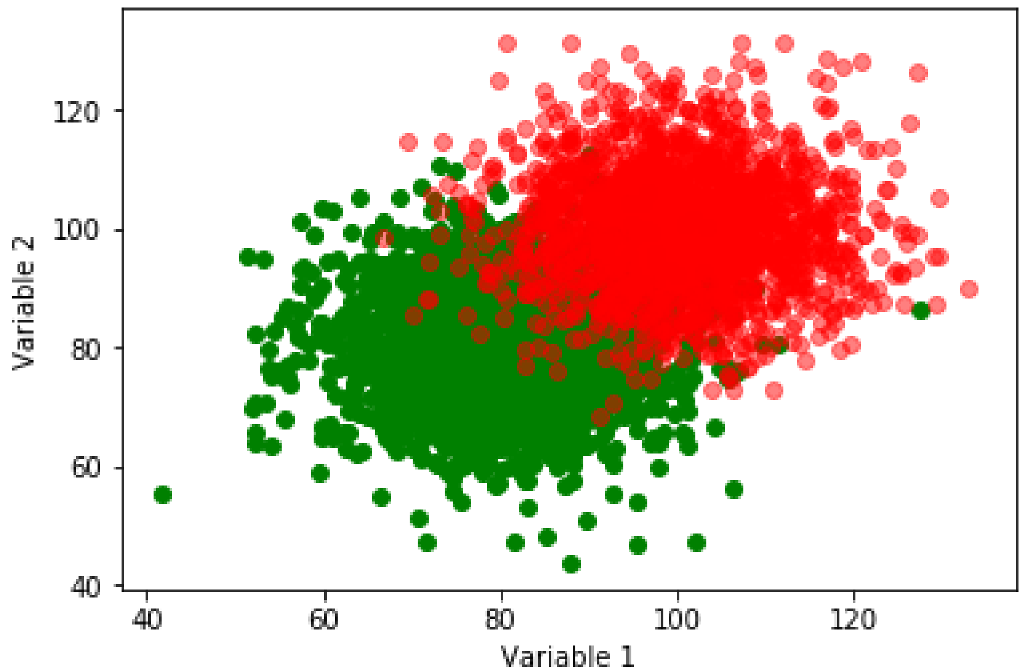
September 14, 2017

In [1]:
```
import sklearn
import pylab
import random
import numpy as np
import pandas as pd
#normal random variates on both x and y dimensions.
## Note that there's no relationship between the two variables


## To get data suitable for displaying on a scatterplot,
## generate normal random variates on both x and y dimensions.
## Note that there's no relationship between the two variables.
```

In [2]:
```
sampleSize = 1000

x3 = []
y3 = []

for i in range(sampleSize):
    x3.append(random.normalvariate(85,10))
    y3.append(random.normalvariate(85,10))


pylab.scatter(x3,y3,c="green",alpha=0.8)
pylab.xlabel("Variable 1")
pylab.ylabel("Variable 2")

x4 = []
y4 = []

for i in range(sampleSize):
    x4.append(random.normalvariate(100,10))
    y4.append(random.normalvariate(100,10))

pylab.scatter(x4,y4,c="red",alpha=0.6)
pylab.show()
```




```
In [3]: sample2Size = 2000

        x5 = []
        y5 = []

        for i in range(sample2Size):
            x5.append(random.normalvariate(80,10))
            y5.append(random.normalvariate(80,10))


        pylab.scatter(x5,y5,c="green")
        pylab.xlabel("Variable 1")
        pylab.ylabel("Variable 2")

        x6 = []
        y6 = []

        for i in range(sample2Size):
            x6.append(random.normalvariate(100,10))
            y6.append(random.normalvariate(100,10))

        pylab.scatter(x6,y6,c="red",alpha=0.5)
        pylab.show()
```



1 Classification

In [4]:
```
# ML imports
from matplotlib.colors import ListedColormap
from pandas.tools.plotting import scatter_matrix
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [5]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

X = []
Y = []

X = np.array(x3)
```

```
Y = np.array(y3)
validation_size = 0.20
seed = 7

X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validati

y_train = Y_train.astype(int)

cmap_light = ListedColormap(['#AAAAFF', '#FFAAAA'])
cmap_bold = ListedColormap(['#0000FF', '#FF0000'])

# fit a SVM model to the data
svm_clf= svm.SVC()

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

svm_clf.fit(X_train.reshape(-1,1), y_train)
```

Out[5]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [19]:
```
# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
#y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5

h = .02  # step size in the mesh
#xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

#Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

print(X_train.shape, y_train.shape)
print(X_validation.shape, Y_validation.shape)
#lookupTable, Z = np.unique(np.array(Z),return_inverse=True)
#Z = Z.reshape(xx.shape)

#fig, ax = plt.subplots()
#plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
#cbar = plt.colorbar()
#cbar.set_ticks([0.25,0.75])
#cbar.set_ticklabels(['ER Prediction Region','NR Prediction Region'])
#cbar.ax.set_yticklabels(cbar.ax.get_yticklabels(),rotation=-90)
#cbar.ax.invert_yaxis()
```

```
# Plot also the training points
#lookupTable, Y_colors = np.unique(np.array(Y_train),return_inverse=True)
#ax.scatter(X_train[:, 0], X_train[:, 1], color=Y_colors, cmap=cmap_bold,
#            edgecolor='k', s=30, alpha=0.1)
#plt.xlim(xx.min(), xx.max())
#plt.ylim(yy.min(), yy.max())
#plt.title("Figure 7: SVM ER/NR Classification Training")
#plt.xlabel('cs1 (PE)')
#plt.ylabel('log10(cs2_bottom/cs1)')
#plt.show()
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-19-720c09ae77d7> in <module>()
  1 # Plot the decision boundary. For that, we will assign a color to each
  2 # point in the mesh [x_min, x_max]x[y_min, y_max].
----> 3 x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
  4 #y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
  5


IndexError: too many indices for array
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.cross_validation import train_test_split
from sklearn.svm import SVR
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e3)
svr_poly = SVR(kernel='poly', C=1e3, degree=3)
y_rbf = svr_rbf.fit(X_train.reshape(-1,1), Y_train).predict(X_train.reshape(-1,1))
#y_lin = svr_lin.fit(X_train, Y_train).predict(X_train)
#y_poly = svr_poly.fit(x3, y3).predict(X)

lw = 2
plt.figure(figsize=(12, 7))
#plt.plot(X, y_rbf, color='navy', lw=lw, label='RBF model')
#plt.plot(X, y_lin, color='c', lw=lw, label='Linear model')
#plt.plot(X, y_poly, color='cornflowerblue', lw=lw, label='Polynomial model')
#plt.xlabel('data')
#plt.ylabel('target')
#plt.title('Support Vector Regression')
```

```
    #plt.legend()
    #plt.show()
```

```
In [ ]:
```